VZDU # 30 MAY/JUNE 1991

DOWN UNDER CLUB

Editor
Harry Huggins
12 Thomas Str.
Mitcham. 3132.
03-873 1408

Treasurer
Ron Allen
2 Orlando str.
Hampton.3188.
03-598 4534

In the last VZDU pages 11 and 12 are reversed. Just turn the page over and switch the numbers. Sorry about that.

Once again we have "lots of goodies". I continue Bob's SOUND series and David's Mystery program writing. Dave Mitchell did'nt get flooded out, and has sent us a program that will formatt (init) one nominated track. I tell you how Ii needed to make use of it almost as soon as I git it.
Paul has turned his attention to commenting on action games.

Aust. Post is not gathering enough sheckles,so have made alterations to their parcel rates. Like Telecom they advise the public a month or so after they become effective.

I suggest you look into the $1 prepaid envelopes, as the most desirable for our use. They take an A4 page folded, can be up to 20 mm thick and can be up to 500grams. That is enough to hold 5 VZDU Newsletters, or 2 cassettes or about 8 disks with stiffening added. As packets that would cost $2.80. They are carried by air and interstate. If you want bigger there is the $2 envelope which is twice the size, but still limited to the 20mm thickness. I receive articles here with twice the postage necessary on them, and it makes my heart bleed.

We are extending our library, by adding a few books. To start there is the VZ200 and VZ300 Tech Manuals, DSE Introduction to computing, DSE First book of programs, DSE Further programming, and DSE Omibus book of programs. As these books cannot be obtained now, we have put a $25 deposit on them, and we think a 1 month loan is reasonable. We would ask thet they be returned earlier if possible as others may want them. Postage. Most will fit the $2 envelope, and that is to your account. We will take it out of the deposit if you choose.
There are a lot more books on hand, and will be listed in the next Newsletter. Almost ready are tapes 7-8-9-10.

Happy Computing,

Harry

SINGLE TRACK FORMATT . . . .

WRITTEN BY DAVE MITCHELL

This program has been a long time comming , it was plagued with bugs. However here is a program that will formatt single tracks.

The program once run will BEEP to let you know it has been loaded from disk. The USR ROUTINE will be set up so you can renter the program if you break out of it.

Displayed on the screen will be :-

SINGLE TRACK FORMATT
BY D.MITCHELL
ENTER TRACK NUMBER :

The TRACK NUMBER is to be ENTERED in HEX . A WORD OF WARNING :- If a track already has data , IT WILL BE LOST IF YOU FORMATT THIS TRACK.

When a track has been formatted the program restarts with the BEEP to let you know it is completed. This program is usefull when data is lost from track zero.

```
00010 FORI=-28672TO-27895:READA:POKEI,A:B=B+A:NEXT
00020 IFB<>80322THENPRINT"ERROR IN DATA":END
00030 IFPEEK(16384)<>170THENPRINT"NO DISK DRIVE":END
00040 BSAVE"FORMATT",9000,9308
00100 DATA33,6,144,34,142,120,243,205,80,52,251,253,126,20,183
00110 DATA40,13,243,245,205,8,64,241,71,205,62,64,205,11,64,33
00120 DATA140,146,205,117,43,205,58,5,218,190,29,26,254,32,202
00130 DATA25,26,35,6,2,17,0,0,126,254,48,56,31,254,58,56,10,254
00140 DATA65,56,23,254,71,48,19,214,7,230,15,72,6,4,203,35,203
00150
DATA18,16,250,179,95,65,215,16,220,120,254,2,210,30,144,123
00160 DATA254,40,48,161,243,245,50,110,146,253,119,18,33,123,146
00170 DATA134,50,112,146,126,50,111,146,253,119,17,205,8,64,241
00180 DATA183,40,4,71,205,59,64,219,19,203,127,40,28,205,11,64
00190 DATA251,33,212,146,205,117,43,205,73,0,243,205,80,52,205
00200
DATA8,64,1,100,0,205,56,64,24,222,1,232,3,205,56,64,253,229
00210 DATA225,17,77,0,25,253,117,14,253,116,15,235,33,99,146,1
00220 DATA24,0,237,176,98,107,54,0,19,1,130,0,237,176,253,54,56
00230 DATA17,1,144,1,205,56,64,253,110,14,253,102,15,17,11,0,25
00240 DATA84,93,19,66,75,3,217,1,100,0,205,56,64,253,126,51,203
00250 DATA183,253,119,51,211,16,1,100,0,205,56,64,221,33,123,146
00260 DATA253,110,14,253,102,15,253,86,51,6,154,205,43,145,253
00270
```

```
DATA114,51,217,221,126,1,221,35,18,134,2,26,217,254,255,194
00280 DATA3,145,205,11,64,251,195,6,144,78,62,32,170,203,17,210
00290
DATA63,145,211,16,238,32,87,43,211,16,195,74,145,211,16,238
00300 DATA0,87,43,211,16,195,74,145,35,195,78,145,195,81,145,219
00310
DATA18,62,32,170,203,17,210,102,145,211,16,238,32,87,43,211
00320 DATA16,195,113,145,211,16,238,0,87,43,211,16,195,113,145
00330 DATA35,195,117,145,195,120,145,219,18,62,32,170,203,17,210
00340 DATA141,145,211,16,238,32,87,43,211,16,195,152,145,211,16
00350 DATA238,0,87,43,211,16,195,152,145,35,195,156,145,195,159
00360 DATA145,219,18,62,32,170,203,17,210,180,145,211,16,238,32
00370 DATA87,43,211,16,195,191,145,211,16,238,0,87,43,211,16,195
00380
DATA191,145,35,195,195,145,195,198,145,219,18,62,32,170,203
00390 DATA17,210,219,145,211,16,238,32,87,43,211,16,195,230,145
00400 DATA211,16,238,0,87,43,211,16,195,230,145,35,195,234,145
00410 DATA195,237,145,219,18,62,32,170,203,17,210,2,146,211,16
00420 DATA238,32,87,43,211,16,195,13,146,211,16,238,0,87,43,211
00430
DATA10,195,13,146,35,195,17,146,195,20,146,219,18,62,32,170
00440
DATA203,17,210,41,146,211,16,238,32,87,43,211,16,195,52,146
00450 DATA211,16,238,0,87,43,211,16,195,52,146,35,195,56,146,195
00460 DATA59,146,219,18,62,32,170,203,17,210,80,146,211,16,238
00470 DATA32,87,43,211,16,195,91,146,211,16,238,0,87,43,211,16
00480
DATA195,91,146,35,35,0,5,194,43,145,201,128,128,128,128,128
00490 DATA128,0,254,231,24,195,0,0,0,128,128,128,128,128,0,195
00500 DATA24,231,254,0,8,1,9,2,10,3,11,4,12,5,13,6,14,7,15,255
00510 DATA31,32,32,32,32,32,32,83,73,78,71,76,69,32,84,82,65,67
00520 DATA75,32,70,79,82,77,65,84,84,13,32,32,32,32,32,32,32,32
00530 DATA66,89,32,32,68,46,77,73,84,67,72,69,76,76,13,69,78,84
00540 DATA69,82,32,84,82,65,67,75,32,78,85,77,66,69,82,32,58,0
00550 DATA13,32,32,32,82,69,77,79,86,69,32,87,82,73,84,69,32,80
00560 DATA82,79,84,69,67,84,32,76,65,66,69,76,13,32,32,32,32,32
00570 DATA32,32,80,82,69,83,83,32,82,69,84,85,82,78,13,0,0,0
```

****
TEST NOISE
** RBK **
** 31/8/90 **
****

```
97 '
98 '***TEST VARIOUS NOISES USING BEEP ROUTINE.
99 '
100 CLS:POKE 30862,80:POKE 30863,52     :'***BEEP ROUTINE.
110 PRINT:PRINT"          VARIOUS NOISES":PRINT:PRINT
120 PRINT" 1. RADIATION DETECTOR"
130 PRINT" 2. HEART BEAT"
140 PRINT" 3. WHATISIT"
150 PRINT" 4. WHATISIT"
160 PRINT:PRINT"ENTER CHOICE 1-4":PRINT:PRINT
170 INPUT"HOLD ANY KEY TO QUIT ";OP$
180 IF OP$="1" THEN GOTO 1000
190 IF OP$="2" THEN GOTO 2000
200 IF OP$="3" THEN GOTO 3000
210 IF OP$="4" THEN GOTO 4000
220 GOTO 100
230 '
1000 '***RADIATION DETECTOR.
1010 IF RND(0) < 0.5 THEN GOTO 1010
1020 X=USR(0)
1030 GOSUB 5000:IF A$=""THEN GOTO 1010 ELSE GOTO 100
1040 '
2000 '***HEARTBEAT.
2010 X=USR(0)
2020 X=USR(0)
2030 X=USR(0)
2040 FOR I%=0 TO 250:NEXT I%
2050 X=USR(0)
2060 FOR I%=0 TO 250:NEXT I%
2070 GOSUB 5000:IF A$=""THEN GOTO 2010 ELSE GOTO 100
2080 '
3000 '***WHATISIT.
3010 FOR I%=0 TO 30
3020    X=USR(0)
3030    X=USR(0)
3040    FOR N%=0 TO I%
3050    NEXT N%
3060 NEXT I%
3070 GOSUB 5000:IF A$=""THEN GOTO 3010 ELSE GOTO 100
3080 '
4000 '***WHATISIT.
4010 FOR I%=0 TO 160 STEP 4
4020    GOSUB 4100
4030 NEXT I%
4040 FOR I%=160 TO 0 STEP -4
```

```
4050    GOSUB 4100
4060 NEXT I%
4070 GOSUB 5000:IF A$=""THEN GOTO 4010 ELSE GOTO 100
4080 '
4100 X=USR(0)
4110 X=USR(0)
4120 X=USR(0)
4130 FOR N%=0 TO I%:NEXT N%
4140 RETURN
4150 '
5000 '***QUIT ROUTINE.
5010 A$=INKEY$:A$=INKEY$
5020 RETURN
5030 '
10000 CLS:PRINT"ERASING NOISE":ERA"NOISE"
10010 PRINT"SAVING NOISE":SAVE"NOISE"
10020 END
```

****

VARY D.C.

****

```
97 '
98 '***VARY DUTY CYCLE OF  SQUARE WAVE.
99 '
100 CLS:POKE 30862,80:POKE 30863,52    :'***BEEP ROUTINE.
110 INPUT" ON TIME 0-255 ";N%
120 PRINT
130 INPUT"OFF TIME 0-255 ";F%
140 PRINT:PRINT:PRINT:PRINT"ENTER CTRL BREAK TO STOP"
150 X=USR(0)
160 FOR I%=0 TO N%:NEXT I%
170 X=USR(0)
180 FOR I%=0 TO F%:NEXT I%
190 GOTO 150
10000 CLS:PRINT"ERASING VARYDC":ERA"VARYDC"
10010 PRINT"SAVING VARYDC":SAVE"VARYDC"
10020 '
10030 REM; IF YOU ARE USING TAPE DON'T PUT IN LINE 10000.
10040 FOR 10010 USE LINE 10050
10050 PRINT"TO SAVE VARYDC,START TAPE IN RECORD MODE"
10060 PRINT"PRESS 'CSAVE' AND ENTER NAME. THEN PRESS RETURN"
```

Disk I/O ERROR

How often have you seen this crop up? Or worse still you type DIR and all you get is -----NOTHING. Or READY.

You type STATUS and you get---you guessed it. NOTHING or READY.

What you will find if you use the Disk Doctor is track00 Sector00 is not addressed.

I wrote a rather clumbsy way to get over that some issues back. If you have taken my advice and saved the tracks of track 00 with either Filesearch or Dave Mitchell's Ex DOS and with the FORMATT program in this issue and either the DOC or the Label program you can save that diskfull of goodies. FORMATT will formatt the track you nominate.

Now a clever piece of deduction on my part, (Sherlock Holmes would be proud of me), I found that I can even go further to saving a disk.

I had a disk, a one only, that I tried to copy with COPY from Disk Tool Kit. It reads ~~as~~ much as the memory will hold and then saves it to the destination disk. With a 300 and 16K expansion that is tracks 00 to 13; 14 to 26; and 27 to 39. There was an I/O error in the third gulp. I looked with DOC and found track 39 sect 04 was NOT ADDRESSED. I could have Formatted that track and copied the rest, but I wanted that track. So, here is the clever part, that probably has been in use for the last 9 years. I got the VZ200 out, did'nt put in Mem.Exp. So I had 6K less DOS. That was enough to read about 2 tracks. It copied to the end of track 37, then gave an error. That was 10 tracks more than I had, but I was greedy. I then put DOC in again. As I looked at each sector I would press M--modify--. Then put in another disk and press RETURN. Answer Y to All alteration made. It would then impress the new disk with the contents of that secto. After doing that 31 times I had the 2 tracks copied, except for the track with the error. Can't do anything about that one. But that is only 126 bytes, and some of them will be spaces.

If the fault was in an earlier track, say 5 or 10, then with the use of DOC the other sectors could be copied and then that track FORMATTed and the disk then copied normally.

Never being satisfied, I'm now asking Dave to make FORMATT choose 1 sector r to Formatt!

(Ed.)

I have heard of one person that has never seen that "Track not addressed error". Well, some people have all the breaks. I've seen more than my share.

(Ed.)

ADVENTURE GAME WRITING

DISPLAYING THE PLAYER'S ENVIRONMENT

BY DAVID WOOD

This section is roughly similar for all adventures, and can virtually be copied straight from another adventure listing.

In the demonstration program, a subroutine is called at 4100. This subroutine simply clears the screen and displays the current strength. You may like to print something like:

```
                    THE MYSTERY OF
                   SPOOKY MOUNTAIN

          TIME REMAINING = 383
          STRENGTH = 83   WISDOM = 42
          ===============================
```

However, you may be restricted by the size of the screen in that you may not be able to display this and all the necessary information.

Following are a few lines specific to the demonstration program, that carry out a few sounds, namely a cat and some mosquitoes, in the rooms that they are required.

Following are the next real steps for the display.

```
40 LL=0
43 IF F(21)=1THEN J$=R$+". IT'S DARK. YOU CAN'T SEE A THING."
44 IF F(21)=1THEN GOSUB4850:J$="":O=0:GOTO200
45 J$=R$+". YOU ARE ": GOSUB 4850:J$=""
```

Line 40 sets up the variable LL to equal zero so when the computer starts printing, it "knows" that the cursor is at the left hand edge of the screen. Lines 43 and 44 come into play when the player is in a dark room without a torch. They place into J$ the message,R$, as the result of the players last action, a full stop to end this message, plus a further message - IT'S DARK. YOU CAN'T SEE A THING. They then call the subroutine at line 4850 to print the contents of J$, then jumps to line 200, so that no further information about the player's surrounds are displayed.

(There are two possible ways for adventure programs to deal with players trying to enter dark areas without a light. Some, such as HAUNTED MANSION and MERKFRUIT LODGE, do not allow the player to enter these areas at all. Others, such as the demo program, allow the player to enter the area, but not to see anything, so those who have mapped the area well will be able to find objects if they know where they are.)

Line 45 comes into effect whenever it is not dark - that is, most of the time. It adds the "YOU ARE " part of the room description to the R$ message. It is then printed by the subroutine at 4850. This subroutine enables everything to be printed on the screen with automatic line wrap around. This means that if a word isn't going to fit on the end of a particular line, it is instead printed on the start of the next one, instead of printing the word split between two lines. The descriptions are printed a lot slower on the screen, but the wrap around feature makes them a great deal easier to read.

```
4850 LS=1:LP=1:FOR I=1 TO LEN(J$)
4855 IF MID$(J$,I,1)=" " ANDLL>EL, PRINT MID$(J$,LP,LS-LP):LP=LS+1
```

```
4860 IF MID$(J$,I,1)=" " ANDLL>EL, LL=I-LS
4861 IF MID$(J$,I,1)=" ",LS=I
4870 LL=LL+1:NEXT I
4880 PRINT MID$(J$,LP,LEN(J$)-LP);
4890 RETURN
```

The subroutine scans through the string J$ for spaces. After each character, the variable LL is increased by one. If it finds a space, it's position in the string is recorded in LS. If the space is found and the value of LL is greater than the width of the screen, the message is printed from the piece of the message that was last printed, up until the previous space, the length of the message that has so far been printed is stored in LP, and the value of LL is altered to the current length that has been scanned minus the position of that previous space. Finally line 4890 prints the rest of the message between the last new line and the end of this message. All of this sounds like rather a mouthful, so we will try an example. We will assume that J$="THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG. YOU ARE IN THE DOG'S KENNEL."

Firstly LL is set at 0 and LS and LP are equal to 1. The message is scanned, with LL initially taking the same value as I. When I is equal to 4, a space is found at the end of "The" so the variable LS (think of these initials standing for Last Space) is altered to equal four. Each time a space is found at the end of the next few words, the value of LS is altered to equal the same value as I, until at the end of the word "over" LS is equal to 30. This is where things begin to happen. At the end of the next word (another "the") a space is found, but the value of LL, which has been steadily increasing has now become greater than EL (which is equal to 31). The part of J$ from the start to the value of the last space - from LP (equal to 1) to LS (equals 30) is printed on the screen. This is from the start of the message up to "over." The value of LP (Last Print) is made equal to that of the last space (30) PLUS one, because we do not want to print the space on the start of the next line. LL is then altered to equal I - LS Which is the length scanned after the end of "over" - in this case, four.

It then continues to scan through the message, increasing LL with each character and making LS equal I with the occurence of each space. Again after the word "dog's" we find that LL has exceed 31, so everything between character number LP (thirty) and LS (fifty-nine) is printed (from the words "the lazy" to "in the"). Again LP is updated to the last space plus one (sixty) and ll is altered to the length of "dog's" (five). After scanning a little more, the end of the message is reached and program flow passes to line 4880. Here everything from the last character printed to the end of the message is placed upon the screen - that is, the words "dog's kennel. " After all of that (no wonder printing stuff on the screen takes such a long time!) we are RETURNed to whatever line we called the subroutine from.

The next lines in the program read the room description data and print it on the screen (using another call to line4850):

```
50 IFO<>R,L=4990+R*10: GOSUB 5300: GOSUB 4800 ELSEJ$=C$
55 GOSUB 4850
60 C$=J$:O=R:J$="":B$="":A$=""
```

Because the data reading subroutine takes a fair amount of time, these lines are set up so that once the room description has first been

read, it is stored in C$, and dosen't have to be read again. If you keep your room descriptions short (one data line only), it would save memory, and probably be quicker, if the data was read every time. Other interesting points about these lines are the use of the restore line number routine at 5300, mentioned in an earlier issue, and the use of a comma instead of the THEN command in line 50. This doesn't save any memory, but may enable you to squeeze a few extra characters into a line.

The data reading subroutine at 4800 follows:

```
4800 READA$:B$=B$+A$
4810 IF RIGHT$(B$,1)<>"0" AND RIGHT$(B$,1)<>"1" THEN 4800
4815 A$="":D$=RIGHT$(B$,4)
4820 J$=Z$( VAL( LEFT$(B$,1))) + Z$(VAL (MID$(B$,2,1))+4)
4830 J$=j$+MID$(B$,3, LEN(B$)-5): RETURN
```

In lines 4800 and 4810, a line of data in the description is read, added on to B$, then the last character of it is checked. If this is a zero or a one, it is part of the movement code, and is therefore the end of the description. If not, the program returns to line 4800 to read the next piece of data.

Line 4815 takes the movement code from the end of B$, and places this in D$ for future use.

Line 4820 "decodes" the first two characters of the description from "15" to "in the " or whatever the code may be. The VAL(LEFT$(B$,1)) command takes the first number's numerical value, so if the first number is three, the string in Z$(3) is added to J$ - in this case "by ". The second part of this command adds the string in Z$(second number +4), so if this second number was 2 the contents of Z$(6) - ("an ") will be added to J$ as well.

Line 4830 adds the rest of the room description in B$ to J$ (except for the movement code) then returns to the main routine.

If your descriptions are all no longer than one line, you can omit line 4810 and alter line 4800 to READ B$. If you wish you can make line 50 unconditional (remove everything up to and including the comma and everything after and including the ELSE statement.), delete line 55, and alter line 60 to GOSUB 4850: J$="":B$="" .This may save some string memory.

Lines 70 to 140 check to see if there are any visible gettable objects in the room.

```
70 FORI=1TOG:IFC(I)<>R OR F(I)<>0 THEN90
80 L=4695+I*5:GOSUB5300: READO$:J$=J$+Z$(ASC(O$)-44)
85 J$=J$+RIGHT$(O$,LEN(O$)-1) +", "
90 NEXT
100 IFJ$<>"",J$=LEFT$(J$, LEN(J$)-2)
140 IFJ$<>"",J$=" THERE IS "+J$+" HERE AND "
```

The loop between lines 70 and 90 goes through all the gettable objects and checks to see if each is both in the room and visible. If this is not the case the program flow goes to line 90 which NEXTs the loop so that the next object is checked. If an object in the room is visible, the line number in which its description occurs, is calculated, restored to, and this data is read. The last statement in line 80 decodes the number at the start of the description, then in line 85 the rest of the description is added on with a comma and a space.

At the end of the loop the last comma and space are removed from J$ and it is converted into a message such as "THERE IS A JOYSTICK, A HAMBURGER HERE AND ". If there is no visible object in the room, this message of course is not created.

Lines 101 to 139 are left blank in case you have any special object which the player can see but is not gettable. Objects such as these are normally just included in the room description, but for objects which may change somewhere in the course of the adventure game you would place them here. For example you may have a seedling that grows into a vicious killer cauliflower, or an animated skeleton (in most adventures these tend to be rather unfriendly!) which the player may decide to hack into stock cubes.

eg

```
IFR=63AND F(98)=0 THEN J$=J$+ "A SEEDLING,"
IFR=63AND F(98)=1 THEN J$=J$+ "A VICIOUS KILLER CAULIFLOWER,"
IFR=87AND F(99)=0 THEN J$=J$+ "AN UNFRIENDLY ANIMATED SKELETON,"
IFR=87AND F(99)=1 THEN J$=J$+ "A PILE OF SKELETON STOCK CUBES,"
```

The next section of the program deal with displaying the possible exits.

```
150 J$=J$+"THERE ARE EXITS TO ":M=0:K=0
156 FORI=1TO4:IFMID$(D$,I,1) ="0",M=M+1:NEXT ELSE NEXT
160 FORI=1TO4:IFMID$(D$,I,1) ="1"THEN180
162 J$=J$+Z$(9+I):K=K+1
165 IF K<>M AND K<>M-1 THEN J$=J$+", "
170 IF K=M-1,J$=J$+" AND "
175 IF K=M, J$=j$+". "
180 NEXT
190 GOSUB4850
```

Line 156 checks to see how many exits there are from the room so the program can work out where to put commas, "and" and the full stop in the sentence. There is then a separate loop which adds the relevant exits to J$. It checks D$ (the movement code) in the order north, south, west, east. If there is a one there is no exit in that direction so the rest of the instructions in the loop are ignored. If there is an exit in that direction, the appropriate direction can be added from the Z$ array (elements 10 to 13 are "north","south","west" and "east" respectively). If this is not the second last or last direction in which there is an exit, a comma is added. After the second last direction the word " and " is added, and after the last, a full stop, so the message may read "there are exits to the north, the south and the east." or "there are exits to the north and the west." or "there are exits to the south."

Finally another call is made to the display subroutine at 4850 which displays all the information about the visible objects and exits.

If you wish, and if you have room, you may like to add another row of equals signs before the "WHAT NEXT?" message

================================

ADVENTURE GAME WRITING - THE COMMAND PARSER

Once again this section is roughly the same for all adventures so you may copy a command parser from any adventure listing you have handy. For those of you who haven't come across the term before, the command parser is the section of the program which accepts the player's input and works out what it all means. Accepting the input is the easy part...

```
195 POKE31060,40:POKE31063,20: FORI=1TO3:X=USR(0):NEXT
197 FORI=1to100:NEXT:POKE 31063,40:FORI=1TO3:X=USR(0): NEXT
200 PRINT:R$="PARDON?":INPUT "WHAT NEXT";A$
```

Lines 195 and 197 are a general prompt sound to let the player know that the computer is finally ready to accept some input (and perhaps wake the player up!). The message "PARDON?" is put into R$ for cases where the input is not filtered out by the error checking routine but doesn't cause a reply from the verb subroutines. This can happen when the player enters two legitimate words which don't make any sense when put together (eg. EXTINGUISH ROCK). The rest of line 200 is fairly self-explanatory.

The next section of the program splits the input up into a verb and a noun.

```
205 IFA$="QUIT"THENCLS:END
210 V$="":T$="":VB=0:B=0
220 FORI=1TOLEN(A$)
230 IFMID$(A$,I,1)=" "AND V$=""THENV$=LEFT$(A$,I-1)
240 IFMID$(A$,I+1,1)<>" "AND V$<>"", T$=MID$(A$,I+1, LEN(A$)-1)
245 IFMID$(A$,I+1,1)<>" "AND V$<>""THENI=LEN(A$)
250 NEXT:IFT$=""THENV$=A$
```

If the player types QUIT the screen is cleared and the program ends. If not, the strings and variables in line 210 are cleared because they contain information from the player's last input which is no longer needed. The loop looks through the input for a space, and when one is found, everything before the space is placed into V$ and everything from the next non-space character onwards is placed in T$. Once this has been done, there is no need to continue the loop so I is set equal to the length of the input, thus ending the loop at the NEXT statement. If nothing is in T$ (the noun string) after the loop, then it is assumed the player has entered a one word input (such as DIG) so the entire contents of the input is placed in the verb string,V$.

```
255 IFV$="GO"THENV$=LEFT$(T$,1)
257 IFV$="NORTH"ORV$="SOUTH" ORV$="EAST"ORV$="WEST",V$= LEFT$(V$,1)
260 IFLEN(V$)<3,V$=V$+"???"
270 IFV$="UNLIGHT"THENV$= "EXTINGUISH"
280 IFA$="DRIVE CAR"THENV$= "START"
```

The first two lines in this part deal with all the different ways the players can enter where they want to go. For example, they could enter GO SOUTH, SOUTH, or just S. These lines convert all the direction commands to single letter format.

Because some commands are less than three letters long and they need to be compared with the three letter strings in X$, three question marks are added to the shorter verbs. (remember that X$= "N??S??W??E??HEL......)

The three letter format can also create some confusion where the

first three letters of different verbs are the same. Line 270 deals with the occurrences of UNLIGHT so they aren't mixed up with UNLOCK.

```
300 U$=LEFT$(T$,3)
310 FORI=1TONV:IF MID$(X$,I*3-2,3)=U$THENVB=I: I=NV
320 NEXT:FORI=1TONO:IF MID$(Y$,I*3-2,3)=LEFT$(T$,3) THENB=I:I=NO
330 NEXT
340 IFT$="DOORMAT"THENB=27ELSE IFT$="BOOKSHELF"THENB=22
```

This part converts the input into a verb number and a noun number. Firstly the first three letters of the player's verb are compared in turn with the three letter parts in X$. If they are the same at any stage, a valid verb has been entered. The verb number is assigned to VB and I is then set to equal the number of verbs to end the loop and save time. Exactly the same process is carried out on Y$ for the nouns. If a valid noun has been entered, its number is placed in B.

Line 340 deals with occurrences of nouns where the first three letters are the same. In this case "DOOR" and "DOORMAT" as well as "BOOK" and "BOOKSHELF".

These lines deal with return messages for input that the computer doesn't understand.

```
355 IFB=0,R$="FOOLING AROUND WITH THE '"+T$+"' WON'T HELP"
356 IFT$=""THENR$="I NEED TWO WORDS"
358 IFVB=0THENR$="I DON'T UNDERSTAND '"+V$+"'"
360 IFVB=0ANDB=0THENR$="THAT'S SILLY!"
370 IFB>GORB=0THEN400
380 IFVB=7OR(VB=24ANDB=9)THEN 400
390 IFVB<>0ANDC(B)<>0THENR$="YOU DO NOT HAVE THE "+T$:GOTO30
```

If the player enters a noun the computer doesn't understand, its reply will be "Fooling around with the '(noun)' won't help". If no noun is entered at all, the reply is "I need two words." however in both cases the computer will still go to the verb subroutine because the noun may not matter, or may not be needed at all (eg. Dig).

If the verb is not understood the message is "I don't understand '(verb)'" and if neither noun or verb is understood the reply is simply "That's silly!" (You can change the error messages if you wish, but make sure the player knows what it is the computer doesn't understand.)

Lines 370 to 390 deal with players trying to do things to gettable objects which they haven't got. Obviously this section doesn't apply to fixed objects so line 370 forces this section to be skipped if the noun is not gettable. Also there are some verbs where the player doesn't have to have objects to do things to them (GET is an obvious example, and there may be others such as a rope that must be tied to the tree before climbing it). line 380 deals with this situation. If the verb is a valid one, and the player is not carrying the object referred to, the message is "You do not have the (object)" and the input is totally rejected.

Finally line 400 (H=R*100+B) is designed to save memory by holding the information about the player's location and object referred to. This means that instead of having to type:

IFR=68ANDB=89THEN.....

You can simply type:

IFH=6889THEN.....

(The last two digits of H are the contents of B and any preceding digits the contents of R.)

GAMES COLUMN

By Paul Frantz

Welcome back to another action-packed edition of the VZDU GAMES COLUMN! Well actually the last couple of months have been fairly slow VZ gamewise but I've still got a bit to say anyway. Firstly I have a suggestion from my brother. "Why don't you review action games instead of adventure games all the time" he said. The answer of course lies in my preference and ongoing search for good VZ games. Still, VZ action games represent a large chunk of the VZ games community and so that's why I'm calling on all those action game freaks to send in their own reviews! That's right here it is, a once-in-a-lifetime opportunity offer to be a substantial part of this very column! If you are interested please contact me at the address below for more info on this rewarding opportunity!

Secondly, there is my VZ PUBLIC DOMAIN TAPE. It would be right to say that my first tape had a bit of a poor start in the world of the eternal VZ and I would like to make an improvement on it (it has taken me ages to get some people's tapes back as well). What I propose is this. A NEW, second tape to hit the Public Domain range. Sounds a bit familiar I hear some of you think, well yes and no. The idea is the same but this time I am offering to all those who contribute to this fine piece of the VZ's future a FREE copy of the completed product i.e. a 90 minute tape jam-packed with VZ programs of all shapes and sizes. It just sounds all so easy doesn't it. Well of course it does rely on you so please make the effort today and send in all those useful little programs and games you have created. Details of the new tapes distribution shall follow next time. Anyway now that's out of my hair we are on to a review with a difference. That's right this time I am not reviewing a true adventure game. Fact is I've about exhausted my collection. This time we have a sneak at a kind of simple strategy game by the name of ...

ESCAPE RIVER

Escape River a.k.a Raft-Away River is by Larry Taylor of LASERLINK fame. This original idea (although I have seen something similar on the Apple II) for a game allows from two to six players to work as a team to save an ill-fated party from rising flood waters.

This situation is this. Your group (in which each member has a different skill e.g. woodchopping) is stranded on the west bank of a fast flowing river. By co-operating and using your appropriate skills, the aim is to work to building a raft to save you from the forthcoming floods. An interesting thing to point out as well is this games use of split-screen mode, a graphics mode enabled with a little assembly programming. This effectively allows the programmer four lines of text at the top of a mode (1) screen. Each action in the game has a pre-assigned key thus saving you the trouble of typing in repetitive commands over and over.

Definately a great game, with a good dose of educational value as well. It is worth seeing even if only to see the good use of the split screen mode. Very well written. This shows just what the poor old VZ graphics chip is capable. The VZ. Still full of surprises!

RATING:
*VOCABULARY: N/A (only suitable criteria for an adventure game)
*PUZZLE DIFFICULTY: 5 (although this can depend on the difficulty chosen.)
*ATMOSPHERE: 8
*LASTABILITY: 7
OVERALL...............6 (and two thirds)

AVAILABILITY: Should still be available for a small sum from Bob Kitch. Address on backpage of VZDU#29.

HOT, HANDY & HELPFUL HINTS

* Yes, some hints! thanks to action game champ Peter Watson for the following: (HIGH SCORE PLAYERS take careful note!)

CRASH: Make sure you keep changing lanes constantly.

KAMIKAZE: Keep moving from the left side of the screen to the right side, according to the side the kamikazes are being released from.

QUESTIONS & QUERIES

? What, no questions? Well yes there are still a couple from last month. I will attempt to delve into them deeply myself. Remember to send your questions as we should have a team of specialised game players ready to help you!

And finally ....

The HIGH SCORES

GAME	SCORE	LEVEL	HOLDER
DAWN PATROL	52500		David Wood
CRASH	581		Peter Watson
DIG OUT	52500		Kenley McLean
HAMBURGER SAM	39500		Stephen Frantz
LADDER CHALLENGE	22530		Peter Watson
KAMIKAZE	36530		Peter Watson
TEN PIN BOWLS	206		Bernice O'Mahoney
VZ INVADERS	30160		Peter Watson
GALAXON	328,460		Mathew McLean
PENGUIN	1350		Jason Oakley
LUNAR LANDER	4600		Ben Hobson
SUPER SNAKE	1918	Novice	Peter Watson
MAZE OF ARGON	73888		Peter Watson
ASTEROIDS	110000		Peter McLean
CIRCUS	1210		Peter Watson
PANIK	7700		Peter Watson
HOPPY	25550		Matthew McLean
GHOST HUNTER	23400		Chris McLean
STAR BLASTER	480 units left		Matthew McLean
KNIGHTS & DRAGONS	3700	Easy	Peter Watson

Well well. Not too much happening here with the exception of Peter Watson (HIGH SCORE TABLE LEADER) who has gone to the trouble of beating three of his own records. In one case he has doubled his old score! (KAMIKAZE). The other point of interest is Ben Hobson who has stolen LUNAR LANDER away from Jason Oakley. Will Jason make a comeback. Find out next time. Right here. Oh yes, watch out for me next time. I'm feeling a bit trigger happy myself. You have been warned!

Anyway that's about it once more. I hope to hear from a lot of you out there concerning a lot of the mini-columns above (especially the empty ones). Please consider the rewarding benefits of contributing to the next chapter in VZ public domain and write to me at my address below. Thank you for your support!

And so as always, see you next edition!

Please send all your hints, high scores, questions and answers, reviews and large cash donations in to:

Paul Frantz
25 Crocker St
KIRWAN QLD 4817

WAVZ 200/300 (computer) Users' Group

By Graeme Bywater.

The great little (so called toy) has done it again. In January 1987 I was offered a computer program to do the "Australian Speedcar Championships". The program was originaly a spread sheet designed by our former foundation president Andre Schoen. His program found its way over to South Australia to a Mr Dodd and altered it to suit the needs of progressive point scoring in use of the up and coming championships At Claremont Speedway. We (as a club) jumped at it in doing the job. So on the 9th and 10th of January, 1987, the successful job was done. The previous system was by hand, this method long and a higher place getter was some times over looked in puting the final placeings and pole positions for the final. This little VZ computer was used for the FIRST time as a method of point scoreing in Australia on any speedway for such an event no other computer (to our knowledge) has been used for this purpose.

So comes 91, and according Speedway's calendar, the "Australian Speedcar Championships are on again on the 1st and 2nd of February, and so the little VZ comes up trumps again.

There were 44 entries from all over Australia, even Northern Territory. of them 10 did not start.

The eventual winners are :- (1)N.T.1 Ekins. (2)W.A.4 Jones. (3)W.A.76 Watson Jnr. (4)N.T.51 Blake. (5)W.A.34 Figiomeni. (6)W.A.75 Watson Snr. (7)W.A.9 Cover. (8)S.A.23 Herreen. (9)W.A.21 staton. (10)W.A.29 Glazebrook.

LET'S INVESTIGATE SOUND ON THE VZ. PART IV

by Bob Kitch.

For the next session on sound generation on the VZ, I will detail some articles on peripheral devices that can be connected to the VZ. These can greatly expand the appeal of the machine and enhance your interest in the VZ. (not to mention the enthusiasm that others will get for the computer.)

There are two types of "noise making" peripherals. These are VOICE and SOUND synthesis I.C. chips. These are alternate and novel forms of output, to that obtained from the screen or printer, when one has tired of these entirely visual forms of output. Music synthesis exceeds the capabilities of the VZ's inbuilt piezo-speaker.

A number of circuits and projects have appeared in the magazines over the past few years. This article briefly identifies these for those who may wish to build a board or alternatively register interest with me so that we can make available these peripherals plus some off-the-shelf software.

Imagine the blockbusting use of voice and music synthesis in games or applications for the VZ.

A couple of introductory articles on speech synthesis appeared in BYTE Sep. 84, p.337 and ITEC #26, p.812. These provide good background.

Magazine	Date	Name	Chip	Interface	Software
VOICE					
EA	Oct. 82 Apr. 83	Compu-voice	Votrax SC-01	Centronics	Yes
APC	Dec. 84	DIY synth.	SC-01	Centronics	Yes
ETI	Jan. 85 Apr. 86	Chatter-box	SC-01	Centronics	Yes
ETI	Mar. 86	Talking VZ-200	GI SP0256	Parallel	No
AEM	Feb. 86	Project 4505	GI SP0256	Centronics	Yes
PE	Mar. 85 Jun. 85	BBC	GI SP0256	Parallel	Yes
PE	Jan. 86	Spectrum			Yes
SOUND					
APC	Nov. 84	DIY synth.	TI SN76496	Centronics	Yes
EA	Aug. 83	Compu-muse	TI SN76489	Centronics	Yes

So if you are tired of reading output from your computer, why not try listening instead?